Lab 4: ARM Assembly Sort

Introduction

In this lab you will learn how to write ARM assembly code to write a simple sorting algorithm.

Learning Objectives

By the end of this lab you will have:

- Written a simple assembly program to sort an array of signed bytes
- Used the debugger in SEGGER Embedded Studio to monitor a location in memory.

Requirements

Write an ARM assembly-language program to sort an array of 12 signed bytes on your MCU.

Resources

- Starter Code from GitHub Repo

Lab 4 Specifications

Lab-specific Specifications

Proficiency

- □ Assembly program correctly sorts an array of signed bytes.
- □ Brief description of sort algorithm implemented.
- □ SEGGER Embedded Studio debugger used to demonstrate that the test array was sorted properly.

□ Assembly code includes line-by-line comments
□ Code is running on MCU connected to SEGGER Embedded Studio for instructor to verify operation with a test case of their choosing.

Excellence

□ Report identifies all possible edge cases for sorting **and** presents testing results to verify that the design performs correctly for all the edge cases you've identified. You can demonstrate this by showing screenshots of the intial (`src`) and final (`dst`) arrays in the watch window or memory viewer.

General Specifications

Proficiency

General Schematic Specifications

- □ All pin names labeled
- □ All pin numbers labeled
- □ Crossing wires clearly identified as junction or unconnected
- □ Neat layout (e.g., clear organization and spacing)
- □ All parts labeled with part number
- □ All component values present

Block Diagram

- □ Block diagram present with one block per SystemVerilog module
- □ Each block includes all input and output signals

HDL & Code Specifications

General Formatting

- □ Descriptive filename (e.g., `lab2_jb.sv`)
- □ Descriptive variable names
- □ Neat formatting (e.g., standard indentation, consistent formatting for variable names (kebab-case/snake_case/camelCase/PascalCase))
- □ Descriptive and clear function/module names

Comments

- □ Comments to indicate the purpose of each function/module

Lab Writeup/Summary

- □ Brief (e.g., 3-5 sentence) description of the main goals of the assignment and what was done.
- □ Explanation of design approach. How did you go about designing and implementing the design?
- □ Explanation of testing approach. How did you verify your design was behaving as expected?
- □ Statement of whether the design meets all the requirements. If not, list the shortcomings.
- □ Number of hours spent working on the lab are included.
- □ Writeup contains minimal spelling or grammar issues and any errors do not significantly detract from clarity of the writeup.
- □ (Optional) List commments or suggestions on what was particularly good about the assignment or what you think needs to change in future versions.

Excellence

General Schematic Specifications

- □ Standard symbols used for all components where applicable
- □ Signals “flow” from left to right where possible (e.g., inputs on left hand side, outputs on right hand side)
- □ Title block with author name, title, and date

HDL & Code Specifications

General Formatting

- □ Name, email, and date at the top of every file
- □ Comment at the top of each source code file to describe what is in it
- □ Clear and organized hierarchy (e.g., deliniation between top level modules and submodules)

Testbenches

- □ Testbenches written for each individual module to demonstrate proper operation
- □ Testbench output included in the report

Lab Writeup/Summary

- □ Writeup is free of spelling and grammar issues

Open specifications in new tab.

Instructions

Project Setup

Set up a project in SEGGER Embedded Studio for your MCU. Once your project is created, download the starter code and put it in your src folder or create a new file inside your src directory titled sort.s and copy in the contents of the starter code.

Finish writing the assembly language subroutine under label main in sort.S to sort 12 signed bytes (the ones on the .byte … line under `arr`). Remember that assembly language code is nearly unreadable without line- by-line comments. If needed, use online reference sheets for ARM Thumb2 Assembly language (like the one linked on the course website) or refer to Chapter 6 of Digital Design and Computer Architecture.

Note: If you are running code on your Nucleo while it is connected to the development board, make sure that you remove the UPduino_+5V jumper to avoid powering the FPGA. Also note that you must have the MCU_+5V jumper connected in order to power the voltage regulators and ensure that the MCU reset signal is pulled high.

Running and Testing Your Code

Build your code and enter a debug session. With the MCU board connected via USB, start a debugging session by going to the debug section of the sidebar and pressing the green arrow in the top of the menu (or by using the keyboard shortcut F5).

Examine the memory location where the array of signed bytes has been loaded using the memory location viewer. The memory viewer is normally located near the bottom of the program window, but if you can't find it, you can access it through the "View" menu in the program toolbar.

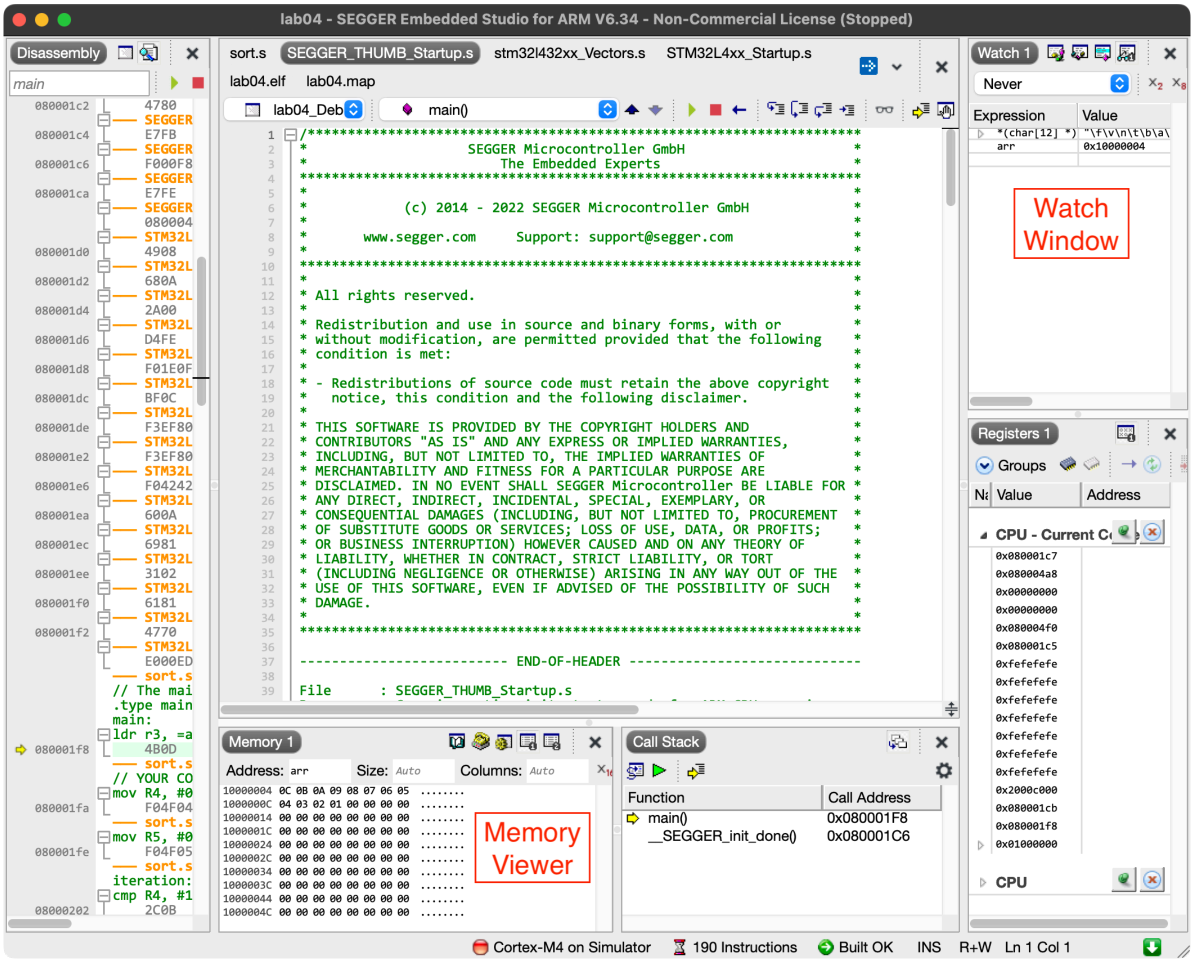
You may also choose to use the "Watch" section of the debugger. For example, if you want to view the contents of the byte array stored in memory, you can enter `*(char[12] *) arr` as a Watch expression. This tells the watch window to cast the address of the arr label to a pointer to a character and dereference 12 consecutive values from that memory location.

Step through the operation of your program and see how it changes the values of the array stored in memory.

To test your code, try various cases with the array in the `.byte...` line under the `arr` label. Rebuild and start a new debug session every time you make changes.

What to Turn In

- A short writeup of your design approach and the sort algorithm you chose to implement.
- A copy of your assembly code file sort.s.
- A listing of the test cases you used and the output of the tests. Be sure your tests would convince a skeptic that your algorithm works.
- For your lab checkoff, please have your lab set up and connected to a running debugger session so that the instructor may provide a test case for your code.
- How many hours did you spend on this lab?



Disassembly: main

```
080001c2     4780
          —— SEGGER
080001c4     E7FB
          —— SEGGER
080001c6     F000F8
          —— SEGGER
080001ca     E7FE
          —— SEGGER
             080004
          —— STM32L
080001d0     4908
          —— STM32L
080001d2     680A
          —— STM32L
080001d4     2A00
          —— STM32L
080001d6     D4FE
          —— STM32L
080001d8     F01E0F
          —— STM32L
080001dc     BF0C
          —— STM32L
080001de     F3EF80
          —— STM32L
080001e2     F3EF80
          —— STM32L
080001e6     F04242
          —— STM32L
080001ea     600A
          —— STM32L
080001ec     6981
          —— STM32L
080001ee     3102
          —— STM32L
080001f0     6181
          —— STM32L
080001f2     4770
          —— STM32L
             E000ED
          —— sort.s
          // The mai
          .type main
          main:
          ldr r3, =a
080001f8     4B0D
          —— sort.s
          // YOUR CO
          mov R4, #0
080001fa     F04F04
          —— sort.s
          mov R5, #0
080001fe     F04F05
          —— sort.s
          iteration:
          cmp R4, #1
08000202     2C0B
```

```
/*********************************************************************
*                    SEGGER Microcontroller GmbH                     *
*                        The Embedded Experts                        *
**********************************************************************
*                                                                    *
*            (c) 2014 - 2022 SEGGER Microcontroller GmbH             *
*                                                                    *
*       www.segger.com     Support: support@segger.com               *
*                                                                    *
**********************************************************************
*                                                                    *
* All rights reserved.                                               *
*                                                                    *
* Redistribution and use in source and binary forms, with or         *
* without modification, are permitted provided that the following    *
* condition is met:                                                  *
*                                                                    *
* - Redistributions of source code must retain the above copyright   *
*   notice, this condition and the following disclaimer.             *
*                                                                    *
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND             *
* CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,        *
* INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF           *
* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE           *
* DISCLAIMED. IN NO EVENT SHALL SEGGER Microcontroller BE LIABLE FOR *
* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR           *
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT  *
* OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;    *
* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF      *
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT          *
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  *
* USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH   *
* DAMAGE.                                                            *
*                                                                    *
**********************************************************************

-------------------------- END-OF-HEADER -----------------------------

File      : SEGGER_THUMB_Startup.s
```

Watch 1: Never

Expression	Value
*(char[12] *)	"\f\v\n\t\b\a\
arr	0x10000004

Registers 1

Na	Value	Address
CPU - Current Co		
	0x080001c7	
	0x080004a8	
	0x00000000	
	0x00000000	
	0x080004f0	
	0x080001c5	
	0xfefefefe	
	0xfefefefe	
	0xfefefefe	
	0xfefefefe	
	0xfefefefe	
	0xfefefefe	
	0xfefefefe	
	0x2000c000	
	0x080001cb	
	0x080001f8	
	0x01000000	
CPU		

Memory 1 — Address: arr Size: Auto Columns: Auto

```
10000004 0C 0B 0A 09 08 07 06 05  ........
1000000C 04 03 02 01 00 00 00 00  ........
10000014 00 00 00 00 00 00 00 00  ........
1000001C 00 00 00 00 00 00 00 00  ........
10000024 00 00 00 00 00 00 00 00  ........
1000002C 00 00 00 00 00 00 00 00  ........
10000034 00 00 00 00 00 00 00 00  ........
1000003C 00 00 00 00 00 00 00 00  ........
10000044 00 00 00 00 00 00 00 00  ........
1000004C 00 00 00 00 00 00 00 00  ........
```

Call Stack

Function	Call Address
main()	0x080001F8
__SEGGER_init_done()	0x080001C6

Figure 1: SEGGER Embedded Studio (SES).